Jeremias Isohanni

ANDROID-SOVELLUKSEN KEHITTÄMINEN

RSS-lukija

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Kesäkuu 2017

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Kesäkuu 2017	**Tekijä/tekijät** Jeremias Isohanni
Koulutusohjelma Tietotekniikka		
Työn nimi ANDROID-SOVELLUKSEN KEHITTÄMINEN. RSS-lukija		
Työn ohjaaja Sakari Männistö		**Sivumäärä** 27+1
Työelämäohjaaja		

Opinnäytetyön tarkoituksena on kehittää Android-sovellus, jonka tarkoituksena on hakea päivityksiä Kokkolan kaupungin tarjoamista kokouksista ja kokousasioista. Android on Googlen julkaisema käyttöjärjestelmä mobiililaitteille. Androidin käyttäminen ja sovellusten kehittäminen on maksutonta, joten mobiilituotteiden valmistat kuten Samsung, Sony ja LG käyttävät laitteissaan Androidia.

Teoriaosuudessa on kerrottu Androidin historiasta, suosiosta ja sovellusten kategorioista ja määristä. Opinnäytetyössä on myös kerrottu, mihin käyttöjärjestelmään Android pohjautuu ja mitä ohjelmointikieliä Androidissa voidaan käyttää. Teoriaosuudessa on myös käyty läpi tärkeitä sovelluksen komponentteja ja Android Studio -ohjelmointiympäristöä. Opinnäytetyössä käydään myös sovelluksen kehittämiseen tarvittavat protokollat ja näiden pohjalta on luotu suunnitelma sovellukselle.

Kaikki sovellukselle annetut vaatimukset toteutuivat ja nämä testattiin toimiviksi. Sovelluksella voidaan hakea Kokkolan kaupungin suodattimen kokousasioita ja kokouksia Android-mobiililaitteilla. Pohdintaosuudessa käydään läpi eri ongelmatilanteita ja niiden ratkaisut sekä sovelluksen laajentamiseen liittyvät ratkaisut.

Asiasanat
Android, Android ohjelmointi, Android studio, kehittäminen, ohjelmointi, RSS, RSS-lukija, XML

ABSTRACT

Centria University of Applied Sciences	**Date** June 2017	**Author** Jeremias Isohanni
Degree programme Information and Technology		
Name of thesis DEVELOPING ANDROID SOFTWARE. RSS reader		
Instructor Sakari Männistö		**Pages** 27+1
Supervisor		

The aim of this thesis was to develop an Android application whose purpose was to retrieve updates for meetings and meeting items offered by the City of Kokkola. Android was published by Google and it is an operating system for mobile devices. Using and developing Android is free so mobile device manufacturers such as Samsung, Sony and LG use Android for their devices.

The theoretical part describes Android's history, popularity and application categories and quantities. The thesis also presents which operating system Android is based on and which programming languages can be used in Android. The theoretical part also describes important application components in Android and briefly describes an integrated development environment called Android Studio. The thesis also presents different protocols needed for the application and creates a design for the application.

All the requirements for the application were met and were tested to be functional. The application can retrieve meetings and meeting items provided by the City of Kokkola for Android mobile devices. The discussion section covers various problem situations with the project and their possible solutions as well as solutions for expanding the application.

Key words
Android, Android programming, Android studio, developing, programming, RSS, RSS-reader, XML

KÄSITTEIDEN MÄÄRITTELY

GUI	Graphical user interface on visuaalinen tapa interaktoida tietokoneen kanssa
SDK	Software development kit on valikoima sovelluskehitystyökaluja
NDK	Android native language development kit on työkalu, joka mahdollistaa esimerkiksi C ja C++ ohjelmointikielen käyttöä Android-sovelluksessa
IDE	Integrated development environment on ohjelmointiympäristö
APK	Android application package on Android-sovellusten tiedostotyyppi
RAM	Random Access Memory on haihtuvaa keskusmuistia, jota käytetään ohjelmien lataamiseen sekä sovellusten suorittamiseen
API	Application programming interface on ohjelmointirajapinta, joka mahdollistaa ohjelmiston kommunikoinnin keskenään
AVD	Android virtual device on laitteen kokopano, joka suoritetaan Android-emulaattorilla.
ART	Android Runtime on sovelluksen suorituksen järjestelmä, jota Android-käyttöjärjestelmä käyttää

TIIVISTELMÄ
ABSTRACT
KÄSITTEIDEN MÄÄRITTELY
SISÄLLYS

1 JOHDANTO 1

2 ANDROID 2
2.1 Android-sovelluksen rakenne 4
2.1.1 Aktiviteetti 4
2.1.2 Intent 6
2.2 Android sovelluksen kehittäminen 7

3 PROJEKTIN SUUNNITTELU 12
3.1 Projektin rajaaminen ja ominaisuudet 12
3.2 Tiedon hakeminen 12
3.3 Suunnitelma 14
3.3.1 Toiminnallisuus ja käyttöliittymä 15
3.3.2 Testaaminen 16

4 OHJELMOINTI 18
4.1 Kehittäminen 19
4.2 Testaaminen 21

5 POHDINTA 27

LÄHTEET 28

KUVAT

KUVA 1. Most popular Google Play app categories in February 2014, by device installs 3
KUVA 2. Aktiviteetin elämänkaari 6
KUVA 3. Esimerkki Android Virtual Device – ohjaimen Android-laitteesta 9
KUVA 4. Android studion API tason valinta. 11
KUVA 5. RSS-suodattimen html-osoitteet 13
KUVA 6. Sovelluksen mallikuva 15
KUVA 7. Sovelluksen kulku. 16
KUVA 8. Lopullinen tuotos 18

TAULUKOT

TAULUKKO 1 Android API-tasot 8
TAULUKKO 2 Kanavaelementin pakolliset elementit 13
TAULUKKO 3 Projektin eri versiot ja niiden ominaisuudet 14
TAULUKKO 4 Testaustaulukko 16
TAULUKKO 5 Täytetty testaustaulukko 21

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on suunnitella ja toteuttaa Android-sovellus. Ajatus tästä työstä syntyi halustani oppia Android-ohjelmointia kehittämällä oma sovellus. Toissijaisena tavoitteena oli dokumentoida oma työprosessi siten, että aiheesta kiinnostutut henkilöt voisivat hyödyntää opinnäytetyöraporttia omassa opiskelussaan. Ohjelmointikielenä toimii Java-ohjelmointikieli ja ohjelmointiympäristönä Android studio.

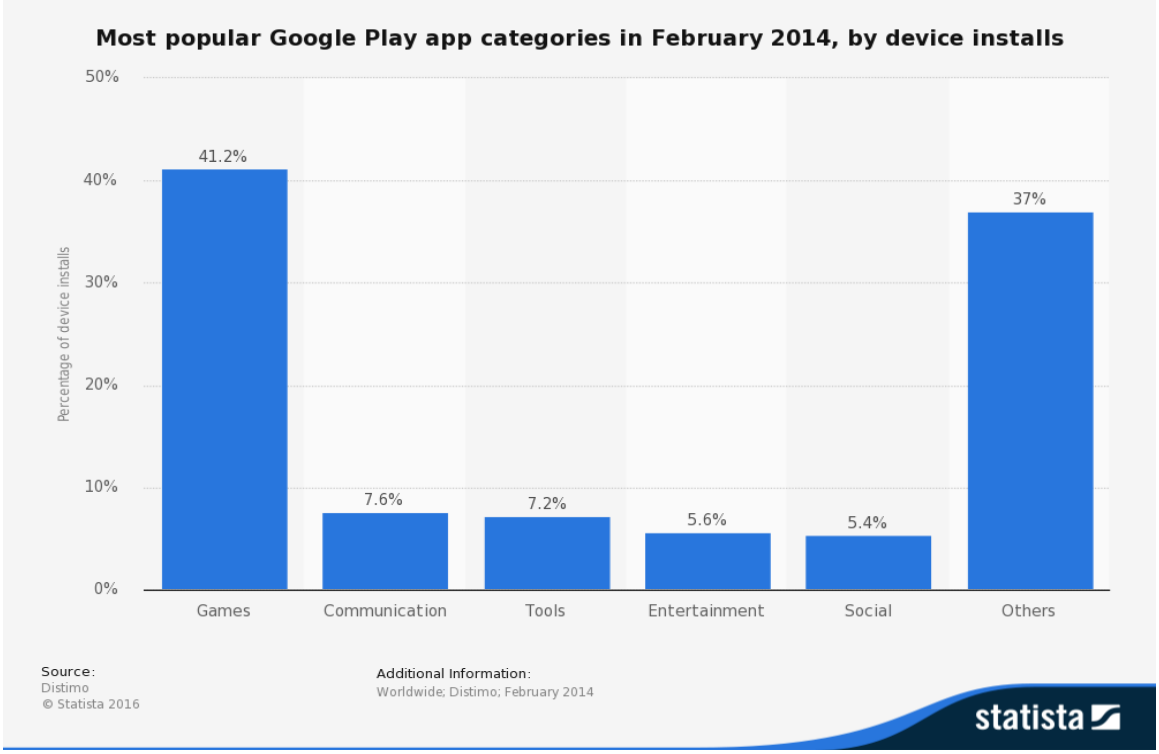
Sovellus tulee olemaan RSS-lukija, jolla käyttäjä pystyy lukemaan kuntien tarjolle laittamaa tietoa kokouksista omalla Android-älypuhelimella. Opinnäytetyön aikana käydään läpi työkalut mitä tällaiseen projektiin vaaditaan ja tullaan käyttämään. Tässä työssä tullaan myös käymään läpi ohjelmointikielet, tietokannat sekä protokollat, joita tarvitaan sovelluksen kehittämiseen, suunnittelemiseen ja toteuttamiseen. Lopuksi vielä testataan lopullista sovellusta ja verrataan tätä alkuperäisiin suunnitelmiin, joista käydään läpi toimivuus ja ulkoasu. Lähteinä käytetään näiden mainittujen työkalujen kehittäjien kirjoittamia ja ylläpitämiä dokumentaatioita. Edellä mainittujen tavoitteiden lisäksi lukijalle esitetään, kuinka haastavaa Android-sovelluksen kehittäminen on ja kuinka alkuperäinen suunnitelma ja lopullinen tuotos poikkeavat toisistaan.

2 ANDROID

Android on käyttöjärjestelmä mobiililaitteille kuten puhelimille ja tableteille. Google julkaisi Android-käyttöjärjestelmän vuonna 2007. Sitä kehitetään avoimena mobiilikäyttöjärjestelmänä Open Handset Alliance -konsortion alaisuudessa, jossa Googlella on johtava rooli. Androidin käyttäminen ja sovellusten kehittämän on maksutonta, koska kyseessä on avoimeen lähdekoodin perustuva alusta ja Google tarjoaa myös ilmaiset kehittämistyökalut. Tämän ansiosta monet valmistajat käyttävät älypuhelimiensa ja mobiililaitteidensa käyttöjärjestelmänä Androidia. Valmistajiin kuuluu muun muassa LG, Samsung, Motorola ja Sony sekä lukuisia pienempiä valmistajia. Android on avoin ohjelmisto, mutta puhelimet yleensä sisältävät Googlen palveluja hyödyntäviä sovelluksia, minkä vuoksi käyttäjällä tulee olla Google-käyttäjätili. Kun Android-sovellusta suunnitellaan ja rakennetaan, on hyvä tietää, että käyttäjällä on pääsy Googlen tarjoamiin palveluihin kuten Google maps, Gmail ja myöskin muihin suosittuihin palveluihin kuten Twitter jotka ovat nykyään lisättynä vakiona Android-älypuhelimiin, mutta tämä tietenkin riippuu kehittäjästä ja puhelimen mallista. (Androidsuomi.fi 2016.)

Koska Android on avointa lähdekoodia ja kehittäminen on maksutonta, useat mobiililaitteiden valmistajat ovat kehittäneet korvaavia sovelluksia ns. vakio-Androidiin. Tämän tarkoituksena on antaa valmistajien omia sovelluksia ja korvata Androidin vakiosovellukset. Androidin vahvuuksiin kuuluu muokattavuus ja käyttäjät voivatkin ladata vaihtoehtoisia sovelluksia valmistajien tarjoamien sovelluksien tilalle. Jokaisessa Android-puhelimisessa on kuitenkin aina tarjottu seuraavia vakiosovelluksia: sähköposti, Facebook, www-selain ja musiikkisoitin. Työpöydällä toimivia sovelluksia kutsutaan widgeteiksi, joita tulee Android-puhelimessa vakiona ja jotkin Android-pohjaiset puhelimet tarjoavat myös flash-sovelluksia. Android-alustalle on saatavilla suuri määrä sovelluksia, joita tarjotaan pääasiassa Google Play -kaupassa. Maksullisia sovelluksia varten tarvitaan Googlen käyttäjätunnus ja voimassa oleva luottokortti. Maksujen hoitamiseen Google on kehittänyt omiin tarkoituksiinsa Google-checkout palvelun, joka muistuttaa tunnettua Paypal-sivustoa. Sovelluksia voi myös ostaa valmistajien omilta sivuilta käyttäen maksuvälineenä esimerkiksi edellä mainittua paypallia. Vaihtoehtoisia kauppoja on kuten Amazonin kehittämä Amazon appstore, mutta tässä opinnäytetyössä keskitytään Googlen omaan Google play storeen. Sovellusten hinnan määrittelee sovelluksen kehittäjä ja hinnat vaihtelevat täysin ilmaisesta sovelluksesta useaan kymmeneen euroon. (Androidsuomi.fi 2016.)

Googlen play kaupassa oli marraskuussa 2015 1 800 000 eri sovellusta ladattavissa joko ilmaiseksi tai maksua vastaan. Sovellusten määrä on kasvanut tasaisesti vuoden 2009 joulukuusta asti, jolloin tarjolla oli vain 16 000 eri sovellusta. Miljoonan eri sovelluksen rajapykälä saavutettiin heinäkuussa 2013. Sovelluksia on lukuisia, joten käyttäjille on tarjolla paljon vaihtoehtoja, millä muokata omaa Android-mobiililaitettaan. Alla olevasta kuvasta voidaan päätellä, että helmikuussa 2014 suosituin Google play kaupan sovellusluokka oli pelit. Toiseksi suosituin sovellusluokka taas oli muut-sovellukset, jotka eivät sovi kuvassa esitettyihin kategorioihin. Kuvassa 1 on myös erikseen eritelty pelit ja viihdekategoriat, vaikka nämä voisi melkein luokitella yhdeksi kategoriaksi, ja samaa väitettä voisi myös käyttää kommunikointi- ja sosiaalisesta-kategorioista. Kuitenkin sovellusten kehittäjän on syytä miettiä minkälaisia sovelluksia käyttäjät pääosin lataavat Googlen play-kaupasta. (Statista.com 2016.)



KUVA 1. Most popular Google Play app categories in February 2015, by device installs (Statista.com. 2016)

Android-käyttöjärjestelmässä on hyvin yksinkertaisesti selitettynä kaksi osaa, jotka pystyvät lukemaan ja suorittamaan sovelluksia. Ensimmäinen osa on Googlen kehittämä Linux-käyttöjärjestelmään perustuva alusta, jota on muokattu mobiililaitteille sopivaksi. Toinen osa pohjautui dalvik-virtuaalikoneeseen, mutta tämä korvattiin myöhemmin Android Runtime (ART) ohjelmalla. Google myös tarjoaa SDK:n ilmaiseksi kenen tahansa käytettäviksi. Tässä työssä tullaan käyttämään Android-

studiota ohjelmointiympäristöä. Muita vaihtoehtoja on Eclipse-pohjainen kehitysympäristö, mutta se vaatii erillisen pluginin Googlelta, jonka avulla voidaan kehittää sovelluksia Android-alustalle. Vaikka aikaisemmin mainittiin niin Android ei kuitenkaan käytä virallista Java-ohjelmointikieltä. Google käyttää apache harmony-luokkakirjastoa. Vaikka kyseessä ei ole virallinen Java-ohjelmointikieli, apace harmony tarjoaa suurin piirtein samat luokkakirjastot kuin virallinen Java. Muita ohjelmointikieliä on myös mahdollista käyttää Android-sovellusten kehittämiseen, kuten esimerkiksi C- tai C++-ohjelmointikielet. C ja C++-ohjelmointikielillä kehitettyjä kirjastoja voidaan käyttää NDK:n rajapintojen kautta. Ehtona tietenkin, että kirjastot ovat saatavilla ja tukevat mobiililaitteiden prosessoreita. (Androidsuomi.fi 2016; Source.android.com.)

2.1 Android-sovelluksen rakenne

2.1.1 Aktiviteetti

Aktiviteetti on sovelluksen komponentti, joka tarjoaa käyttäjälle jonkin toiminnon ja siihen liittyen käytttöliittymänäkymän. Aktiviteetti voi käyttää koko puhelimen näytön tai pientä osaa siitä, vaikka yleensä aktiviteetti vie koko näytön tilan. Yleensä jokaisessa sovelluksessa on useampi aktiviteetti-ikkuna, jotka ovat liitettynä toisiinsa esimerkiksi valikkojen kautta. Sovelluksissa on aina yksi pää-aktiviteetti, joka suoritetaan aina sovelluksen alkaessa. Jokainen aktiviteetti voi aloittaa toisen aktiviteetin, jolla voi suorittaa tiettyjä komentoja. Kun aktiviteetti aloitetaan, niin aikaisempi aktiviteetti pysäytetään ja järjestelmä pistää pysäytetyn aktiviteetin pinoon odottamaan vuoroaan. Pino on tietorakenne, joka sijoittaa datan pinoon first-in last out metodilla. Eli kun useampi aktiviteetti pysäytetään ja lisätään pinoon, niin se aktiviteetti joka lisättiin pinoon ensimmäisenä, palautetaan näytölle viimeisenä. Tämän ansiosta voimme tehdä muutoksia toisessa aktiviteetissa ilman, että se vaikuttaa ensimmäiseen aktiviteetin. Kun käyttäjä painaa takaisin-nappulaa mobiililaitteessaan, nykyinen aktiviteetti tuhotaan ja päällimmäinen pinossa oleva pysäytetty aktiviteetti palautetaan laitteen näytölle. (Developer.Android.com. 2016.)

Aktiviteetin pysäytyksen yhteydessä samalla kun toinen aktiviteetti suoritetaan, ilmoitetaan pysäytetylle aktiviteetille tilan muutoksesta aktiviteetin elinkaarimetodeilla. Metodeja on useampi, ja niiden käyttö ja kutsuminen riippuvat aktiviteetin tilan muutoksesta. Järjestelmä voi muuttaa aktiviteetin tilaa joko luomalla aktiviteetin, pysäyttämällä aktiviteetin, palauttamalla aktiviteetin ja tuhoamalla aktiviteetin.

Jokainen näistä tason muutoksista antaa kehittäjälle mahdollisuuden tehdä jotakin taustalla. Esimerkiksi aktiviteetin pysäytyksen aikana kehittäjä voi vapauttaa resursseja, kuten pysäyttämällä yhteyden tietoverkko- ja tietokanta-yhteyden. Kun aktiviteetti taas palautetaan voi aktiviteetti palauttaa käyttöön pakolliset resurssit ja palauttamalla toiminnot, jotka keskeytettiin aktiviteettistä poistuessa. (Developer.Android.com. 2016.)

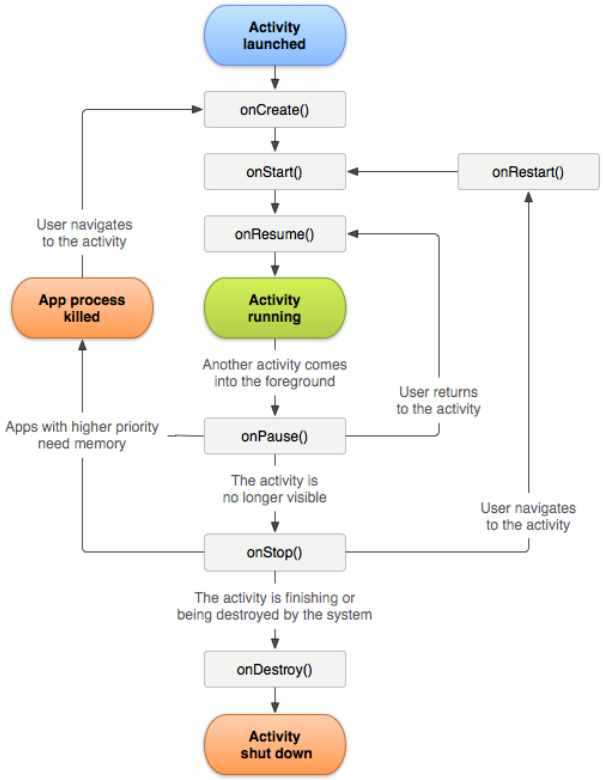
Aktiviteetin elinkaaren hallinta on tärkeää ottaa huomioon sovellusta kehittäessä. Aktiviteetin elinkaareen vaikuttavat suoraan toiset aktiviteetit, aktiviteetin tehtävä ja pino tietorakenne. Aktiviteetillä on kolme eri tilaa:

- Suorituksessa: Aktiviteetti on mobiililaitteen näytössä päällimmäisenä ja käytettävissä. Tästä tilasta voidaan käyttää myös nimityksiä aktiivinen tai palautettu.
- Pysäytetty: Aktiviteetti on taustalla, ja jossakin tapauksissa tämä ikkuna näkyy myös käyttäjälle. Tämä tarkoittaa sitä, että toinen aktiviteetti ei peitä mobiililaitteen koko näyttöä tai ikkuna on läpinäkyvä. Pysäytetty aktiviteetti on kuitenkin suorituksessa ja käyttää laitteen resursseja. Aktiviteetti-objekti pitää siis muistin, kaikki tilamuutokset, jäseninformaation ja pysyy lisättynä ikkunanhallintaohjelmassa. Järjestelmä voi kuitenkin sammuttaa aktiviteetin, jos järjestelmän välimuisti on loppumassa.
- Lopetettu: Aktiviteetti on taustalla ja on täysin toisen aktiviteetin peitossa. Lopetettu aktiviteetti on silti suorituksessa ja käyttää laitteen resursseja kuten pysäytetty aktiviteetti. Toisin kuin pysäytetty aktiviteetti, lopetettu aktiviteetti ei pysy lisättynä ikkunanhallintaohjelmassa. Lopetetun aktiivin tunnukset ovat, että se ei enää näy käyttäjälle ja se voidaan sammuttaa, jos välimuistia tarvitaan muualla laitteessa.

Jos aktiviteetti on pysäytetty tai lopetettu, järjestelmä voi sammuttaa sen kutsumalla aktiviteetin sammuttamisen metodia finish() tai sitten sammuttaa sen tappamalla prosessin. Kun käyttäjä suorittaa aktiviteetin, joko sammuttamisen tai prosessin tappamisen jälkeen, aktiviteetti täytyy luoda uudestaan. (Developer.Android.com. 2016.)

Aktiviteetin elinkaari on nyt käyty läpi, joten voidaan todeta, kuinka tähän elinkaareen voidaan vaikuttaa. Kun aktiviteetti muuttaa tilaa palautetun, pysäytetyn ja lopetetun välillä, se ilmoitetaan takaisinkutsu-metodilla. Takaisinkutsu-metodit ovat koukkuja, jotka voivat muuttaa sopivia tiloja aina kun aktiviteetin-tila muuttuu sovellusta käytettäessä. Metodit ovat jaettu kolmeen eri silmukkaan. Näitä silmukoita voidaan monitoroida ja kuva 2 kuvastaa tämän elinkaaren. Aktiviteetin elinkaari alkaa, kun onCreate()-metodia kutsumalla. OnCreate()-metodissa on hyvä luoda aktiviteetin layout ja muut globaalit muuttujat aktiviteetille. Aktiviteetin näkyvä elinkaari taas alkaa kutsusta onStart() ja jatkuu

onResume() metodiin asti. Tämän jälkeen käyttäjä voi vuoro vaikuttaa aktiviteetin kanssa ja tehdä muutoksia aktiviteettiin. OnPause()-metodia kutsutaan esimerkiksi, kun mobiililaitteen näyttö pimennetään tai dialogi ilmestyy aktiviteetissa. OnStop()-metodia kutsutaan kun toinen aktiviteetti peittää aikaisemman aktiviteetin kokonaan ikkunasta. Lopuksi onDestroy()-metodi tuhotaan elinkaaressa luodut toiminnot ja muuttujat. Tärkeintä tässä elinkaaressa on käyttää vain kevyitä toimintoja, koska muuten muutos aktiviteetistä toiseen voi olla hidasta. (Developer.Android.com. 2016.)



KUVA 2. Aktiviteetin elämänkaari (Developer.Android.com. 2016)

2.1.2 Intent

Intent on keskustelu-objekti sovelluksen eri komponenttien välillä. Sovelluksen komponentit voivat intentin avulla pyytää toisen sovelluskomponentin toiminnon käynnistymistä. Intentillä on useita käyttöjä Androidissa, mutta meille tärkeimmät toiminnot ovat aktiviteetin aloittaminen, palvelun aloittaminen ja tiedon kuljettaminen aktiviteetista toiseen. Tämä mahdollistaa tiedon liikuttamisen aktiviteettien välillä, koska intent-objekti kuvastaa aktiviteetin aloituksen ja siirtää kaiken tarpeellisen datan mukanaan uuteen aktiviteettiin. Palvelu on komponentti, joka suorittaa komentoja taustalla ilman että käyttäjä näkee tätä. Intentillä voimme aloittaa uuden palvelun ja kuten aktiviteetin aloittamisessa

intent-objekti kuvaa palvelun aloittamisen ja siirtää kaikki tarvittavat tiedot uuteen palveluun. Lähetys on taas viesti, jonka voi mikä tahansa android-järjestelmä itsessään tuottaa ja lähettää useita lähetyksiä ja viestit vaihtelevat järjestelmän käynnistyksestä mobiililaitteen akun lataamiseen. (Developer.Android.com. 2016.)

On olemassa kahta eri intent-tyyppiä. Täsmälliset intentit kertovat komponentin nimeltä, jonka aktiviteetti haluaa käynnistää. Tämä on yleisesti projektin sisällä oleva luokka-tiedosto. Tämän takia täsmällistä intenttiä käytetään projektin sisällä käynnistämään jokin tietty aktiviteetti tai palvelu. Esimerkkinä, kun käyttäjän toimintojen takia käynnistetään uusi aktiviteetti tai taustalla oleva palvelu. Epäsuora intentti taas ei nimeä suoraan mitään komponenttia, vaan nimeää toiminnon, jonka objekti haluaa suoritettavan. Tämä mahdollistaa komponentin suorittamisen toisessa sovelluksessa. Esimerkkinä, jos sovellus haluaisi näyttää käyttäjälle sijainnin kartalla, mutta ei itse pysty tähän voi sovellus lähettää epäsuoran intentin. Toinen sovellus voi vastata tähän pyyntöön ja näyttää käyttäjälle sijainnin kartalta. (Developer.Android.com. 2016.)

Kun sovellus luo täsmällisen intentin, jolla pyritään aloittamaan aktiviteettiä tai palvelua, niin järjestelmä suorittaa intentissä nimetyn komponentin ja pyrkii suorittamaan tämän heti. Epäsuora intentissä taas Android järjestelmä pyrkii etsimään sopivaa komponenttia. Järjestelmä tekee tämän vertaamalla alkuperäistä intenttiä eri intent-suodattimiin, jotka ovat nimetty manifest-tiedostoissa mobiililaitteen toisissa sovelluksissa. Jos sopiva sovellus löytyy niin järjestelmä käynnistää sovelluksen tai jos sopivia sovelluksia on useita, järjestelmä antaa käyttäjälle mahdollisuuden valita dialog-ikkunan avulla. Intent-suodatin on sovelluksen kehittäjän tekemä suodatin, jossa kehittäjä on määritellyt minkälaisia intent-komponentteja sovellus haluaa vastaanottaa. Kehittäjä voi suodattaa intent-suodatinta esimerkiksi antamalla toiselle sovellukselle oikeuden käynnistää aktiviteetin hänen sovelluksessa. Tietoturvallisuus on tietenkin huomioitava ja jos et halua, että toinen sovellus voi käynnistää toimintoja sinun sovelluksessa voit jättää ilmoittamatta komponentteja sovelluksesi intent-suodattimeen. Tämän seuraukseni vain täsmälliset intentit voivat käynnistää komponentteja sovelluksessasi. (Developer.Android.com. 2016.)

2.2 Android sovelluksen kehittäminen

Kun Android-sovellusta kehitetään niin ensimmäiseksi sovelluksen kehittäjän täytyy valita API-taso. API-tasoja on kirjoittamisen hetkellä markkinoilla 23 eri tasoa. Nämä tasot julkaistaan yleensä uuden

sukupolven mobiililaitteiden kanssa. Sovelluksen suunnittelussa ja kehittämisessä pitää ottaa huomioon eri API-tasot, koska kuluttajilla on käytössään eri API-tasojen laitteita. Tämä tuo uusia haasteita sovelluksen kehittämisessä, koska kehittäjä voi valita kaikista uusimman API-tason, mutta tämä myös tarkoittaa että kaikki Android mobiililaitteet eivät voi suorittaa sovellusta. Jokaisessa API-tasossa on siis omat vaatimuksensa ja ominaisuutensa, joita kehittäjä voi hyväksikäyttää sovelluksessansa. Alla olevasta taulukosta ilmenee API-tasojen Android-versio, julkaisupäivä, API-taso ja nimi. Taulukossa on myös mainittu vain julkaisut jossa on uusi nimi, joten esimerkiksi versioita 4.3 (API-taso 18) ja 3.2 (API-taso 13) ei ole mainittu. (Developer.xamarin.com. 2016.)

TAULUKKO 1. Android API-tasot

Android Versio	**Julkaisupäivä**	**API-taso**	**Nimi**
Android 6.0	Elokuu 2015	23	Marshmallow
Android 5.0	Marraskuu 2014	21	Lollipop
Android 4.4W	Kesäkuu 2014	20	Kitkat Watch
Android 4.4	Lokakuu 2013	19	Kitkat
Android 4.1 – 4.1.1	Kesäkuu 2012	16	Jelly Bean
Android 4.0 – 4.0.2	Lokakuu 2011	14	Ice Cream Sandwich
Android 3.0.x	Helmikuu 2011	11	Honeycomb
Android 2.3 – 2.3.2	Marraskuu 2010	9	Gingerbread
Android 2.2.x	Kesäkuu 2010	8	Froyo
Android 2.0	Marraskuu 2009	5	Eclair
Android 1.6	Syyskuu 2009	4	Donut
Android 1.5	Toukokuu 2009	3	Cupcake
Android 1.0	Lokakuu 2008	1	Base

Android-studio on Googlen tarjoama virallinen ohjelmointiympäristö (IDE) Android-alustalle. Android studio on ilmainen ja kuka tahansa voi ladata tämän ja aloittaa sovellusten kehittämisen Android-mobiililaitteille. Ohjelmointiympäristö perustuu Java-pohjaiseen kehitysympäristöön. Android-studio on suunniteltu helpottamaan kehittämistä Android-alustalle ja tarjoaa käyttäjäystävällisen ympäristön. Intellij lisäksi Android studio tarjoaa esimerkiksi seuraavia ominaisuuksia: joustavan gradle-pohjaisen build-ympäristön, build-variaatioita ja useita vaihtoehtoja apk-tiedoston generointiin, ohjelmointi-pohjia jotka ovat suunniteltu auttamaan kehittäjiä rakentamaan yleisiä piirteitä sovellukseensa ja editorin vedä & pudota kehittämiseen. Android-studio näyttää oletuksena projektin tiedostot *Android Project* -

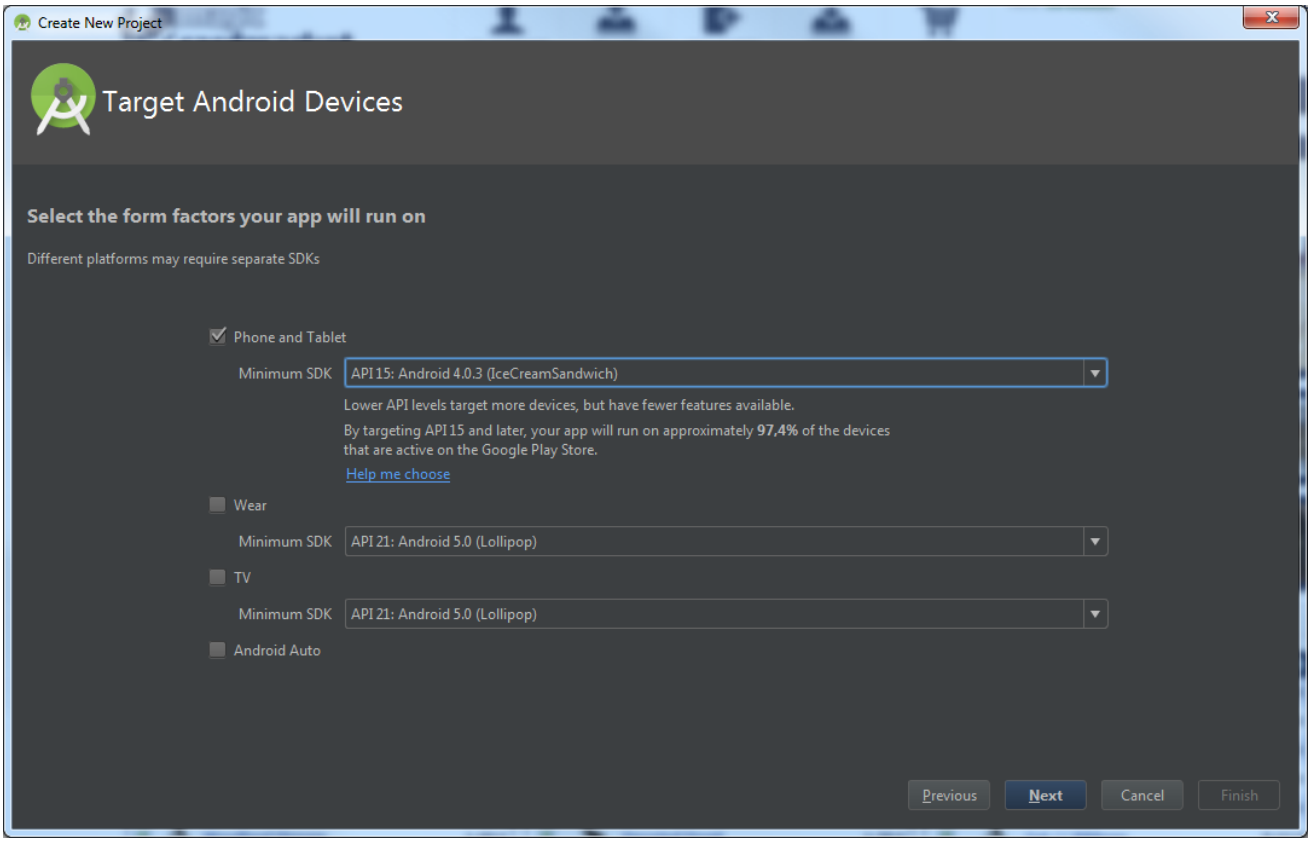
näkymänä, jonka tarkoitus on näyttää kehittäjälle tärkeimmät tiedostot selkeästi ja nopeasti. (Developer.Android.com. 2016.)

Android-studion mukana tulee Androidille tarkoitettu oma koontityökalu. Tämän tarkoitus on antaa kehittäjälle mahdollisuus rakentaa, testata, suorittaa ja pakata projekteja. Koontityökalun saa käyttöönsä Android-studion valikoista tai koontityökalun voi myös suorittaa itsenäisesti komentotulkilla. Android-studio tarjoaa myös mahdollisuuden testata kehitettyä sovellusta joko tietokoneeseen yhdistetyllä fyysisellä Android-mobiililaitteella tai emuloimalla virtuaalisen ympäristön Android Virtual Device (AVD)-ohjaimella. AVD-ohjain luo uusia virtuaalisia ympäristöjä ja kehittäjä voi testata sovellustaan, vaikka fyysistä Android-mobiililaitetta ei olisi saatavilla. Emulaattori luo ulkoasun laitteella, jonka arvo vakiona on Nexus 6 tai Nexus 9 Android-mobiililaite. Kaikki Android-laitteiden toiminnot toimivat näissä virtuaalisissa ympäristöissä. Lyhyesti sanottuna, että AVD-ohjain luo Android-laitteen ohjelmointiympäristölle. Kuvassa 3 on otettu kuvankaappaus virtuaalisessa ympäristössä, joka on muotoiltu Nexus 5 Android-älypuhelimen muotojen mukaan. Tämän avulla kehittäjä pystyy testaamaan, kuinka sovellus toimisi fyysisessä laitteessa. (Developer.Android.com. 2016.)

KUVA 3. Esimerkki Android Virtual Device -ohjaimen Android-laitteesta

Asennuksen yhteydessä Android-studio muokkaa asetuksia käyttöjärjestelmän ja tehokkuuden mukaan. Android-studio tarkistaa onko työympäristössä jo valmiiksi asennettu Java Development Kit (JDK) ja tarkistaa vapaan muistin (RAM). Tämä optimoi Android-studion asetuksia, kuten asetukset liittyen AVD-ohjaimen emulaatioon. Asennuksen jälkeen voit käyttää SDK-ohjaimia työkalujen ja muiden osien päivittämiseen. Android-studiossa on myös neljä eri kanavaa päivityksille, jotka antavat lisää joustavuutta kehittämiseen. Stable-kanava on vakio asennuksen jälkeen ja tämän tarkoitus on päivittää Android-studio vasta kun uusin versio on vakaa ja valmis julkaistattavaksi. Beta-kanava päivittää Android-studion, kun tarjolla on beta-versio tulevasta päivityksestä. Canary-kanava päivittää Android-studion aina kun uusi päivitys on saatavilla melkeinpä viikoittain. Canary-kanavat ovat usein virheellisiä ja niitä ei suositella sovelluksen kehityksen aikana. Dev-kanavat taas ovat Android-studion kehittäjien valitsemia canary-kanavan päivityksiä, jotka ovat olleet tarpeeksi vakaita julkaisuun. Dev-kanavia päivitetään joka toinen viikko tai kuukausittain. (Developer.Android.com. 2016.)

Android-studio antaa myös mahdollisuuden valita, mikä tulee olemaan projektin vanhin tuettu SDK. Tämä helpottaa kehittäjiä valitsemaan heidän projektilleen sopivan SDK:n. Kuvassa 4 on juuri aloitettu uusi projekti ja Android-studio kysyy, mikä on vanhin tuettu SDK-projektille. Tässä tapauksessa se on API15: Android 4.0.3 (IceCreamSandwich) ja kuten näemme Android studio ilmoittaa, että tämä SDK-taso kattaa peräti 97,4% kaikista aktiivisista laitteista Google play kaupan tietojen mukaan. Android-studio myös huomauttaa, että vanhemmilla API-tasoilla saavuttaa enemmän laitteita, mutta niissä on vähemmän ominaisuuksia kuin uudemmissa.



KUVA 4. Android-studion API tason valinta

3 PROJEKTIN SUUNNITTELU

Tässä luvussa käyn läpi projektin suunnitelman eri vaiheet. Kiinnostuin Kokkolan kaupungin tarjoamista esityslistoista ja pöytäkirjoista ja huomasin, että Kokkolan kaupunki tarjoaa näitä esityslistoja ja pöytäkirjoja Kokkolan kaupungin www-sivulla. Huomasin myös, että Kokkolan kaupunki tarjoaa nämä kokousasiat ja kokoukset RSS-suodatuksena. Kehitän siis itselleni RSS-lukijan Android-puhelimelleni, jolla pystyisi lukea näitä kokouksia ja kokousasiakirjoja helposti ja nopeasti. Pohdin myös, että voisiko tätä laajentaa myös muille kaupungeille. Haluan kuitenkin, että sovelluksessa on vähintään mahdollisuus lukea Kokkolan kaupungin kokousasioita ja kokouksia. Helppokäyttöisyys ja sovelluksen ulkonäkö ovat suositeltavia.

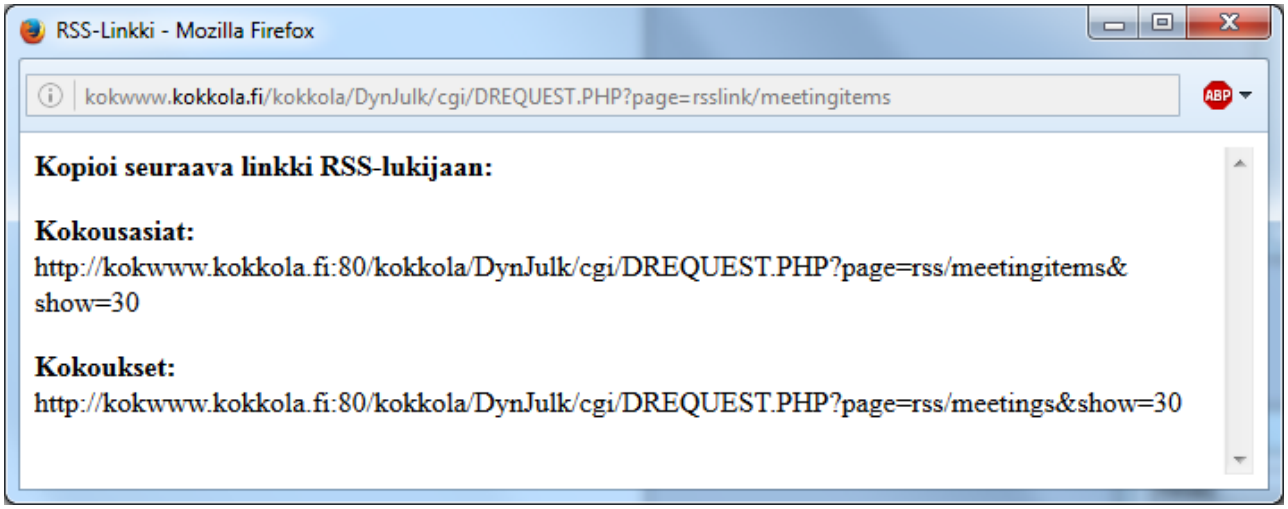
3.1 Projektin rajaaminen ja ominaisuudet

Antamillani ohjeistuksilla voin aloittaa nyt sovelluksen suunnittelemisen. Toiveeni oli tuotaa RSS-lukija Android-älypuhelimelle, jolla luetaan Kokkolan kaupungin julkaisemia kokousasiakirjoja. RSS-lukija on siis tämän sovelluksen päätarkoitus, ja siihen tulen keskittymään tässä työssä. Haluaisin myös mahdollisimman helppokäyttöisen ja hyvän ulkonäön sovellukselle. Nämä kaksi asiaa tullaan pitämään tärkeimpänä ominaisuutena, kun lähden suunnittelemaan sovellusta. Viimeisempänä ominaisuutena voidaan pitää sovelluksen laajentamista muihin paikkakuntiin. Tässä työssä tulen mainitsemaan tästä asiasta, kuinka lisätä muita suodattimia. Rajaan siis sovelluksen siten, että työn päätteeksi minulla on valmis prototyyppi RSS-lukijasta, jolla on mahdollisuus lukea Kokkolan kaupungin esityslistoja ja pöytäkirjoja. Tämän päätoiminnon lisäksi prototyypissä olisi mahdollisuus lisätä muita RSS-suodattimia tai ohjelmointi vaiheessa lisätään mahdollisuus, että tulevaisuudessa pystyn lisäämään RSS-suodattimia helposti. Ominaisuuksia tulisi tämän rajaamisen jälkeen olemaan Kokkolan kaupungin RSS-suodattimen luku, mahdollinen muiden RSS-suodattimien lisääminen ja helppokäyttöisyys.

3.2 Tiedon hakeminen

Nyt minulla on mahdollisuus aloittaa tiedon hakeminen. Koska Kokkolan kaupunki tarjoaa minulle RSS-linkit suoraan, niin on tämä vaihe melko helppo minulle. Kuvasta 5 näen suorat html-osoitteet näille

kahdelle eri suodattimelle ja voin käyttää näitä omassa projektissa. Tästä näen, että voin tarjota työssä vaihtoehtona, joko kokousasioita tai itse kokouksia tai kummatkin suodattimet samanaikaisesti. Tämä on hyvä pitää muistissa, kun alan suunnitella sovelluksen ulkonäköä ja ohjelman kulkua. Nyt kun minulla on informaatio, mistä saan tiedot, voin alkaa käydä läpi, kuinka RSS-suodatin toimii ja teknologiaa ja protokollia tulen tarvitsemaan omassa sovelluksessani.



KUVA 5. RSS-suodattimen html-osoitteet

RSS (Really Simple Syndication) on www-sisällön suodatin, joka käyttää XML-protokollaa. Kaikkien RSS-tiedostojen täytyy olla XML 1.0 -hyväksyttäviä. Korkeimmalla tasolla RSS-dokumentti on osa <rss>-elementtiä, johon täytyy lisätä määriteversio, jota rss-elementti tulee käyttämään. Tämän <rss>-elementin seurana on <channel>-elementti, joka sisältää informaatiota kanavasta (metadata) ja sen sisällöstä. Kanavaelementissä on useita pakollisia ja vapaaehtoisia elementtejä sekä erillinen elementti nimeltään <item>. Alla olevassa taulukossa on mainittu kaikki pakolliset elementit <channel>-elementtiin. Tulen käyttämään näitä elementtejä hyväksemme, kun kehitän omaa sovellusta. RSS-syötteestä on vielä hyvä tietää, että se liittyy XML-kieleen. Tämä tarkoittaa sitä, että kaikki RSS-tiedostot täytyy olla XML 1.0 spesifikaation mukaisia. (Rssboard.org. 2009).

TAULUKKO 2. Kanavaelementin pakolliset elementit

Elementti	Kuvaus	Esimerkki
Otsikko	Kanavan nimi	Uutisotsikko Ry

Linkki	Verkkosivun URL-osoite, joka liittyy kanavaan.	http://www.uutisotsikko.com
Kuvaus	Kanavaan liittyvä kuvaus	Uutisotsikko tuo viimeisimmät uutiset internettiin.

3.3 Suunnitelma

Kuten aikaisemmin kappaleessa mainitsin, haluan RSS-lukijan Android-älypuhelimelle, jolla pystyisi lukemaan Kokkolan kaupungin kokousasiakirjoja ja kokouksia. Alla olevassa taulukossa 3 käyn läpi kolme erilaista versiota sovellukseen. Nämä versiot ovat minimaalinen-, keskiverto- ja täydellinen versio sovelluksesta. Tässä opinnäytetyössä tein minimaalisen version, mutta tulen käymään läpi, kuinka ominaisuuksia muista versioista voisimme toteuttaa projektissa.

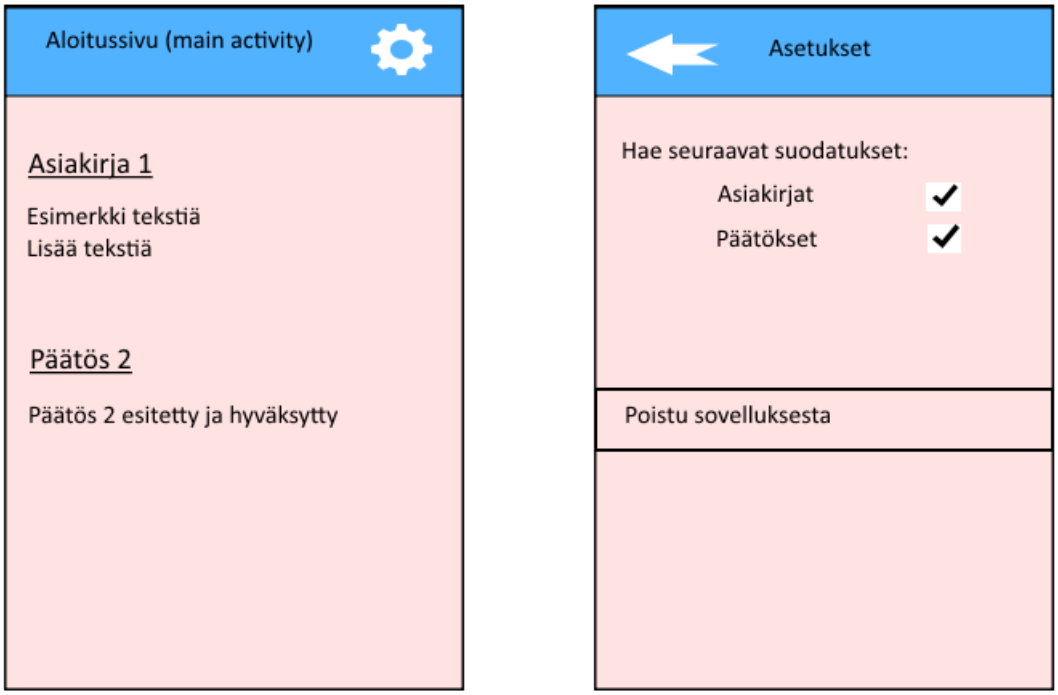
TAULUKKO 3. Projektin eri versiot ja niiden ominaisuudet

	Minimaalinen	**Keskiverto**	**Täydellinen**
Asiakirjojen luku ja poistaminen	x	x	x
Kokouksien luku ja poistaminen	x	x	x
Helppokäyttöinen	x	x	x
Muiden kaupunkien lisääminen ja poistaminen		x	x
Kielen vaihtaminen (Ruotsi ja Englanti)			x

Nyt kun minulla on käsitys mitä haluan sovellukselta, voin aloittaa sen kehittämisen. Ensimmäiseksi esittelen, miten käyttäjä liikkuu sovelluksen sisällä. Tämän pohjalta luon käyttöliittymästä mallikuvan, jota voin käyttää apuna koodauksessa. Viimeisenä käyn läpi mitä erilaisia testimenetelmiä, jolla tulen testaamaan sovellusta.

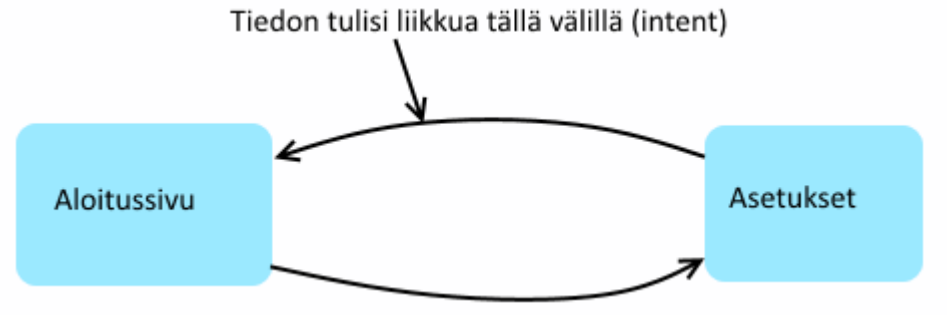
3.3.1 Toiminnallisuus ja käyttöliittymä

Tässä kappaleessa luon sovellukselle kaksi auttavaa kuvaa, joihin tulen palaamaan kehittäessä sovellusta. Ensimmäiseksi luon mallikuvan, jonka pohjalta kehitän sovelluksen älypuhelimelle. Taulukon 3 mukaan tarvitsen ainakin seuraavat ominaisuudet: kokousasioiden luku ja poistaminen, kokouksien luku ja poistaminen ja helppokäyttöisyys. Näistä voin päätellä, että sovelluksessa täytyy olla ainakin 2 aktiviteettia: yksi aktiviteetti kokousasioiden tai kokouksien lukemiseen ja toinen aktiviteetti asetuksille, kuten kokousasioiden ja kokouksien lisäämiselle tai poistamiselle ja muille mahdollisille asetuksille (esimerkiksi sovelluksesta poistuminen). Kuvassa 6 on tekemäni mallikuvasta tähän sovellukseen, jota tulen referoimaan sovelluksen kehittämisessä.



KUVA 6. Sovelluksen mallikuva

Nyt kun minulla on mallikuva valmiina, voin luoda sovelluksen. Kuvasta 6 näen, että käyttäjän ei tarvitse liikkua kuin kahden aktiviteetin välillä (minimalisessa versiossa). Tärkein ominaisuus on asetuksissa tehtyjen muutoksien vaikuttaminen sovelluksen aloitussivuun. Kuvassa 7 on tekemäni piirustus liittyen sovelluksen kulkuun, jota tulen referoimaan sovelluksen kehittämisessä. Tärkeintä on kuitenkin välittää käyttäjän valitsemat asetukset takaisin aloitussivulle.



KUVA 7. Sovelluksen kulku

3.3.2 Testaaminen

Ennen kun aloitan sovelluksen kehittämisen, luon testaamiseen liittyvän taulukon. Ensimmäiseksi mietin, minkälaisia yhteyksiä tai ongelmatilanteita sovelluksessa saattaa syntyä. Toiseksi ajattelin, kuinka käyttäisin sovellusta käyttäjänä, jolla ei ole mitään kokemusta sovelluksesta. Ongelmatilanteita ja virheitä saattaa ilmetä yksinkertaisimassakin sovelluksessa. Tässä vaiheessa käyn myös läpi laitteet, joilla testaan sovelluksen. Puhuin aikaisemmin AVD-ohjaimesta ja tämän avulla voin suorittaa sovelluksen monessa eri laitteessa ilman, että omistan kyseistä laitetta. Laitteiksi olen valinnut Samsung Galaxy S3 - ja Nexus 6 -älypuhelimen (virtuaalinen). Tulen viittaamaan tähän taulukkoon myöhemmin sovellusta kehittäessä ja testaamisessa. Käyn tämän taulukon läpi myöhemmin ja lisään tarvittaessa vielä enemmän tähän taulukkoon, jos uusia ominaisuuksia ilmenee sovellusta kehittäessä.

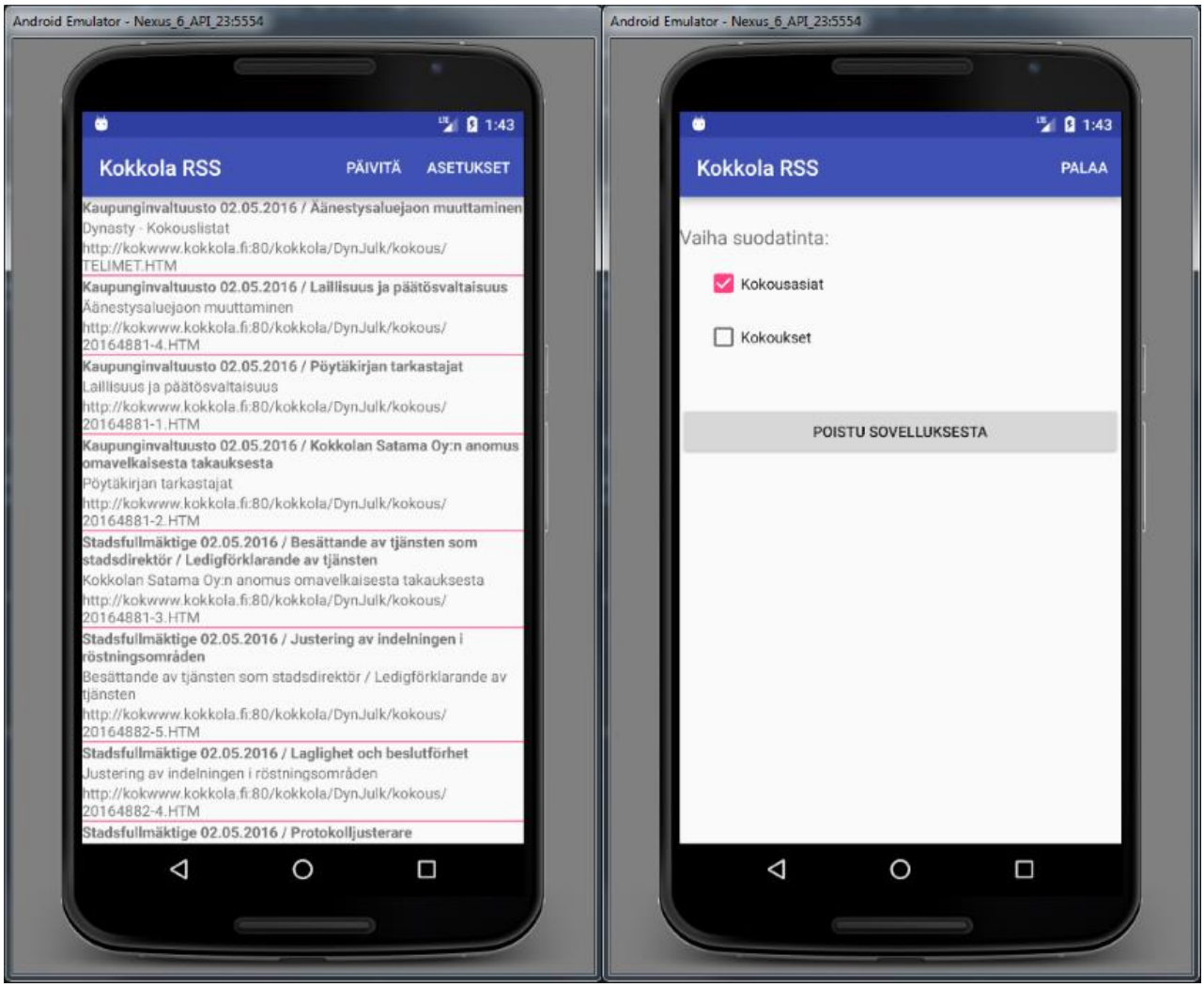
TAULUKKO 4. Testaustaulukko

		Samsung Galaxy S3	**Nexus 6 (AVD)**
1	Aloitussivun tekstit sopivat näytön ruutuun		
2	Asetussivun tekstit sopivat näytön ruutuun		
3	Aloitussivua pystyy vierittää alas ja ylöspäin näytöllä.		
4	Napit ja valintaruudut toimivat aloitussivulla ja asetussivulla.		
5	Aloitussivulta tulee päästä asetussivulle		
6	Asetussivulta tulee päästä aloitussivulle		

7	Asetussivulla tehdyt muutokset tulee näkyä aloitussivulla		
8	Vain kokousasiat näkyvät aloitussivulla		
9	Vain kokoukset näkyvät aloitussivulla		

4 OHJELMOINTI

Viimeisessä luvussa tulen käymään läpi sovelluksen ulkonäön ja kuinka sain toteutettua tärkeimmät ominaisuudet sovelluksen kannalta. Ohjelmoinnissa käytin Android-studio-ohjelmaa ja testilaitteina oli Samsung Galaxy S3 älypuhelin sekä Nexus 6 älypuhelin, jota käytin AVD-ohjaimen kautta. Minulla oli siis käytössä sekä fyysinen Android-älypuhelin ja virtuaalinen versio Android-älypuhelimesta. Kuvassa 8 on valmis lopputulos. Tästä kuvasta ilmenee sovelluksen ulkonäkö sekä kaikki aktiviteetit. Verratakseni tätä sovelluksen mallikuvaan kuvassa 7 näemme, että sovelluksessa ilmenee suurimmaksi osaksi kaikki mallikuvassa tehdyt suunnitelmat. Joitakin erilaisuuksia kuitenkin ilmenee mallikuvaan verrattuna lopulliseen tuotokseen ja käyn eroavaisuudet läpi.



KUVA 8. Lopullinen tuotos

4.1 Kehittäminen

Ensimmäiseksi loin aloitussivun (main activity), johon RSS-syöte tulee. Tähän tarvitsin kolme eri tekstikenttää ja widgetin nimeltä <SwipeRefreshLayout> ja <RecyclerView>. <SwipeRefreshLayout> ja <RecyclerView> lisäävät aktiviteettiin kaksi tärkeää ominaisuutta. Ensimmäiseksi se antaa mahdollisuuden rullata aktiviteettia ylös ja alas. Toiseksi jos aktiviteetissa rullataan aivan ylös ja jatketaan rullaamista niin aktiviteetti päivittää tekstin. Alla olevassa koodissa on tiivistettynä activity_main.xml-tiedoston sisältö.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    xmlns:app="http://schemas.Android.com/apk/res-auto"
    xmlns:tools="http://schemas.Android.com/tools"
    Android:layout_width="match_parent"
    Android:layout_height="match_parent"

    <TextView
        Android:id="@+id/titleFeed"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
    />

    <Android.support.v4.widget.SwipeRefreshLayout
        Android:id="@+id/swipeRefreshLayout"
        Android:layout_width="match_parent"
        Android:layout_height="match_parent"
        >
        <Android.support.v7.widget.RecyclerView
            Android:id="@+id/recyclerView"
            Android:layout_width="match_parent"
            Android:layout_height="match_parent" />
    </Android.support.v4.widget.SwipeRefreshLayout>
</RelativeLayout>
```

Toiseksi loin asetuksille oman aktiviteetin. Tähän aktiviteettiin tarvitsin yhden tekstikentän, kaksi valintalaatikkoa ja yhden napin. Tämän kappaleen alla on tiivistettynä otos activity_settings.xml-tiedoston sisällöstä.

```
?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"

    <TextView
        Android:id="@+id/feedTitleText"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:layout_alignParentLeft="true"/>

    <CheckBox
        Android:id="@+id/meetingItemsCheckbox"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
```

```
        Android:layout_alignParentLeft="true"
        Android:layout_alignParentStart="true"
        Android:checked="false"
        Android:enabled="true"
        Android:onClick="checkBoxClicked"
        tools:enabled="true" />

    <Button
        Android:id="@+id/exitButton"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:onClick="exitProgram"
        />

</RelativeLayout>
```

Kummallakin aktiviteetilla on myös oma valikkonsa, koska ne eivät ole täysin samanlaisia. Valikot toimivat siten, että aktiviteetti kutsuu tiettyä xml-tiedostoa ja näin valikko luodaan. Alle olen lisännyt tiivistettynä valikoitua koodia MainActivity.java- ja app_menu.xml-tiedostoista, joista ilmenee valikon luonti.

MainActivity.java:

```
public boolean onCreateOptionsMenu(Menu menu){
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.app_menu, menu);
    return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case R.id.reloadMenuText:
            //Tähän mitä tapahtuu kun valikon nappulaa painetaan
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

app_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:Android="http://schemas.Android.com/apk/res/Android"
    xmlns:app="http://schemas.Android.com/apk/res-auto">
    <item Android:id="@+id/reloadMenuText"
        app:showAsAction="always"
        Android:title="@string/reloadMenuText" />
</menu>
```

Seuraavaksi on vielä yksi aktiviteetti, joka auttaa suodattamaan RSS-syötteen. Tätä aktiviteettia kutsutaan, kun haemme lähteestämme RSS-syötettä. Aktiviteetin tarkoitus on tulostaa aloitussivuun (Main Activity) suodatettu sisältö. Tähän aktiviteettiin tarvitaan kolme tekstikenttää (RSS title,

description ja linkki) ja yhden <View> erottajan, jotta syötettä on helpompi lukea. Alla on tiivistettynä ote item_rss_feed.xml-sisällöstä.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:orientation="vertical" Android:layout_width="match_parent"
    Android:layout_height="wrap_content">

    <TextView
        Android:id="@+id/titleText"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:textStyle="bold" />

    <View
        Android:layout_width="match_parent"
        Android:layout_height="1dp"
        Android:background="@color/colorAccent" />

</LinearLayout>
```

Viimeisenä mainitsen lyhyesti, mitä täytyy lisätä projektin AndroidManifest.xml- ja build.gradle (module: app) -tiedostoihin. Ensimmäiseksi lisäsin yhden lauseen AndroidManifest.xml-tiedostoon, jonka avulla sovellus pääsee internettiin.

```
<uses-permission Android:name="Android.permission.INTERNET" />
```

Toiseksi muokkasin ja lisäsin seuraavat lauseet build.gradle (module:app)-tiedostoon. Nämä tekevät sen, että RecyclerView toimii sovelluksessa.

```
compile 'com.Android.support:appcompat-v7:23.4.0'
compile 'com.Android.support:recyclerview-v7:23.4.0'
compile 'com.Android.support:design:23.4.0'
```

4.2 Testaaminen

Testaamisessa käytetään taulukko 4 pohjana. Käyn läpi kaikki siinä olevat kohdat ja lisään relevantit kohdat lähdekoodista.

TAULUKKO 5. Täytetty testaustaulukko

		Samsung Galaxy S3	**Nexus 6 (AVD)**
1	Aloitussivun tekstit sopivat näytön ruutuun	**x**	**x**

2	Asetussivun tekstit sopivat näytön ruutuun	**x**	**x**
3	Aloitussivua pystyy vierittää alas ja ylöspäin näytöllä.	**x**	**x**
4	Napit ja valintaruudut toimivat aloitussivulla ja asetussivulla.	**x**	**x**
5	Aloitussivulta tulee päästä asetussivulle	**x**	**x**
6	Asetussivulta tulee päästä aloitussivulle	**x**	**x**
7	Asetussivulla tehdyt muutokset tulee näkyä aloitussivulla	**x**	**x**
8	Vain kokousasiat näkyvät aloitussivulla	**x**	**x**
9	Vain kokoukset näkyvät aloitussivulla	**x**	**x**

Taulukko 5 osoittaa testatut ominaisuudet ja niiden toimivuuden sovelluksessa.

1. Minun ei tarvinnut tehdä paljoa, että sain tämän hoidettua. Activity_main.xml-tiedostossa annoin tekstikentille arvot: ”Android:layout_width=”match_parent” ja tämä kattaa tämän kohdan.
2. Tämä toimii samalla tavalla kuin aikaisempi kohta. Tein vain muutokset activity_settings.xml-tiedostoon.
3. Aikaisemmin lisäsin activity_main.xml-tiedostoon <RecyclerView> widgetin ja tässä kohdassa hyödynsin sitä. Kun koodissa loin RSS-syötteelle pidikkeet, niin seuraavalla koodin pätkillä levitin aktiviteetin ulkonäköä sen verran kuin syötteessä on tietoa.

```
public class RssFeedListAdapter extends RecyclerView.Adapter<RssFeedListA-
dapter.FeedModelViewHolder> {

    private List<RssFeedModel> _RssFeedModels;

    public class FeedModelViewHolder extends RecyclerView.ViewHolder {
        private View rssFeedView;

        public FeedModelViewHolder(View v) {
            super(v);
            rssFeedView = v;
        }
    }

    public RssFeedListAdapter(List<RssFeedModel> rssFeedModels) {
        _RssFeedModels = rssFeedModels;
    }

    @Override
    public FeedModelViewHolder onCreateViewHolder(ViewGroup parent, int
type) {
```

```
        View v = LayoutInflater.from(parent.getContext()).inflate(R.lay-
out.item_rss_feed, parent, false);
        FeedModelViewHolder holder = new FeedModelViewHolder(v);
        return holder;
    }

    @Override
    public void onBindViewHolder(FeedModelViewHolder holder, int position) {
        final RssFeedModel rssFeedModel = _RssFeedModels.get(position);
        ((TextView)holder.rssFeedView.findViewById(R.id.ti-
tleText)).setText(rssFeedModel.title);
        ((TextView)holder.rssFeedView.findViewById(R.id.description-
Text)).setText(rssFeedModel.description);
        ((TextView)holder.rssFeedView.findView-
ById(R.id.linkText)).setText(rssFeedModel.link);
    }

    @Override
    public int getItemCount() {
        return _RssFeedModels.size();
    }
}
```

4. Loin ensimmäiseksi aloitussivun valikot. Nämä kaksi olivat ”Päivitä” ja ”Asetukset”. Kävin jo aikaisemmin läpi, miten menu luodaan aktiviteettiin. Lisäsin seuraavat lauseet aloitussivun koodiin, jotta pystyin toteuttamaan tämän. Case R.id.reloadMenuText päivittää RSS-syötteen ja case R.id.settingsMenuText siirtää käyttäjän asetussivulle.

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case R.id.reloadMenuText:
            new grapFeed().execute((Void) null);
            return true;
        case R.id.settingsMenuText:
            Intent intent = new Intent(this, settingsActivity.class);
            startActivity(intent);
            finish();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Asetussivulle taas on enemmän valintoja ja nappeja. Seuraavassa koodinpätkässä loin muuttujat kahdelle valintalaatikolle ja yhdelle napille sekä listenerit näille. Tämän jälkeen loin uudet funktionit näille valinnoille.

```
private Button exitButton;
private CheckBox meetingItemsCheckbox;
private CheckBox meetingsCheckbox;

  protected void onCreate(Bundle savedInstanceState) {

  exitButton = (Button) findViewById(R.id.exitButton);
  meetingItemsCheckbox = (CheckBox) findViewById(R.id.meetingItemsCheckbox);
```

```
meetingsCheckbox = (CheckBox) findViewById(R.id.meetingsCheckbox);


exitButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
                exitProgram();
        }
});

meetingItemsCheckbox.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
                checkBoxClicked(meetingItemsCheckbox);
        }
});

meetingsCheckbox.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
                checkBoxClicked(meetingsCheckbox);
        }
});

public void exitProgram() {
Intent intent = new Intent(getApplicationContext(), MainActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
intent.putExtra("EXIT", true);
startActivity(intent);
finish();
}

public void checkBoxClicked(View view) {
boolean checked = ((CheckBox) view).isChecked();

SharedPreferences settings = getSharedPreferences("feedSettings", 0);
SharedPreferences.Editor editor = settings.edit();

switch (view.getId()){
        case R.id.meetingItemsCheckbox:
        if(checked){
                meetingsCheckbox.setChecked(false);
                feedSelection = "true";
                editor.putString("feedSelection", feedSelection);
    }
        break;
        case R.id.meetingsCheckbox:
        if(checked){
         meetingItemsCheckbox.setChecked(false);
         feedSelection = "false";
         editor.putString("feedSelection", feedSelection);
    }
        break;
}
editor.apply();
}
```

Viimeiseksi loin paluu-napille koodinpätkän, jolla käyttäjä palaa takaisin aloitussivulle.

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case R.id.returnActivityText:
            Intent intent = new Intent(this, MainActivity.class);
            startActivity(intent);
            finish();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

5. Päästäksemme asetussivulle aloitussivulta lisäsin seuraavan koodinpätkän, kun asetus-nappulaa painetaan.

```
Intent intent = new Intent(this, settingsActivity.class);
startActivity(intent);
finish();
```

6. Sama periaate kuin 6. kohdassa, mutta tällä kertaa lisäsin asetussivun palaa-nappulaan seuraavan koodinpätkän.

```
Intent intent = new Intent(this, MainActivity.class);
startActivity(intent);
finish();
```

7. Tämä onnistui käyttämällä lisäämällä kumpaankin Java-tiedostoon SharedPreferences-muuttuja. Käyttämällä tätä pystyin luomaan yhteisen asetuksen jokaiselle aktiviteetille ja lisäämään sinne arvoja. Seuraavassa koodinpätkässä on esimerkki, kuinka käytin tätä toimintoa hyväkseni.

```
private boolean feedEnabled;

SharedPreferences settings = getSharedPreferences("feedSettings", 0);
SharedPreferences.Editor editor = settings.edit();

if(b) {
    feedSelection = "true";
    editor.putString("feedSelection", feedSelection);
} else if(!b){
    feedSelection = "false";
    editor.putString("feedSelection", feedSelection);
}
editor.apply();

feedEnabled = Boolean.valueOf(settings.getString("feedSelection",""));
```

8. Käyttämällä aikaisempaa SharedPreferences-toimintoa hyväksi loin boolean muuttujan, joka ollessaan totta syöte tulostaa kokousasiat ja epätodessa oltuaan syöte tulostaa kokoukset. Käytin yksinkertaista if-lausetta tähän tehtävään.

```
feedEnabled =  Boolean.valueOf(settings.getString("feedSelection",""));

if(feedEnabled == true) {
    if (TextUtils.isEmpty(urlLinkOne))
        return false;
    try {
        if (!urlLinkOne.startsWith("http://") && !urlLinkOne.starts-
With("https://"))
            urlLinkOne = "http://" + urlLinkOne;

        URL url = new URL(urlLinkOne);
        InputStream inputStream = url.openConnection().getInputStream();
        feedmModelList = parseFeed(inputStream);
        return true;
    } catch (IOException e) {
        Log.e(TAG, "Error", e);
    } catch (XmlPullParserException e) {
        Log.e(TAG, "Error", e);
    }
}
```

9. Sama kuin kohdassa yhdeksän, mutta feedEnabled muuttuja on tällä kertaa epätosi.

```
if(feedEnabled == false) {
    if (TextUtils.isEmpty(urlLinkTwo))
        return false;
    try {
        if (!urlLinkTwo.startsWith("http://") &&
!urlLinkTwo.startsWith("https://"))
            urlLinkTwo = "http://" + urlLinkTwo;

        URL url = new URL(urlLinkTwo);
        InputStream inputStream = url.openConnection().getInputStream();
        feedmModelList = parseFeed(inputStream);
        return true;
    } catch (IOException e) {
        Log.e(TAG, "Error", e);
    } catch (XmlPullParserException e) {
        Log.e(TAG, "Error", e);
    }
}
```

En poikennut paljon alkuperäisestä suunnitelmasta, mutta kuitenkin käyn nämä muutokset lyhyesti läpi. Ensimmäiseksi en nähnyt syytä kokousten tai kokousasiakirjojen poistamiselle, ja näin ollen jätin sen lopullisesta sovelluksesta pois. Toiseksi en luonut käyttäjälle mahdollisuutta lisätä toisten kaupunkien RSS-syötteitä. Kolmanneksi alkuperäisessä suunnitelmassa aloitussivulla asetusvalikkoon siirtymistä kuvasi hammasratas-ikoni oikealla ylänurkassa. En myöskään lisännyt nuoli-ikonia asetussivun vasempaan ylänurkkaan. Tulin siihen päätökseen, että luettavat napit olivat selkeämpiä ja helpommin ymmärrettävissä kuin ikonit. Viimeiseksi en myöskään lisännyt erillistä otsikko-kenttää eri syötteille, ja näin syötteestä tuli helpommin luettavaa.

5 POHDINTA

Kun aloitin opinnäytetyön tekemisen, minulla oli jonkin verran kokemusta Android-ohjelmoinnista. En ollut aikaisemmin tutustunut syventävästi, kuinka Androidin hierarkia toimii, ja tämä oli ensimmäinen kerta, kun kehitin RSS:n tapaisen sovelluksen. Omaksi onnekseni Androidista löytyy paljon eri dokumentaatiota, johdantoja ja esimerkkejä, joten sovellusta tehdessä tai tiedon etsintään ei tarvinnut käyttää paljoa aikaa. Vakuutin itselleni projektin aikana, että pystyn etsimään tietoa ja toteuttamaan sen omaan sovellukseen. Mielestäni sovellus onnistui siinä mihin sen suunnitelinkin ja tähän olen erittäin tyytyväinen. Ongelmatilanteita minulla oli muutama, ja huomattavin näistä oli, että en saanut kumpaakin RSS-syötettä samanaikaisesti näkymään ruudulle. Mielestäni kummankin kokouksen ja kokousasioiden tulostaminen samalle näytölle vei liian paljon tilaa ja toisti itseään. Joten tein sen päätöksen, että käyttäjä voi tulostaa vain toisen syötteen.

Sovellus itsessään on melko rajallinen ja toteuttaa tällä hetkellä vain muutaman toiminnon. Esimerkiksi sovellusta laajentaessa koodia täytyisi muuttaa, että se kattaisi useamman RSS-syötteen. Tällä hetkellä sovellus kattaa vain kahta eri RSS-syötettä. Parantaakseni sovellusta ensimmäiseksi antaisin käyttäjälle mahdollisuuden lisätä ja poistaa minkä tahansa RSS-syötteen. Toiseksi yrittäisin hankkia kaikkien niiden kuntien RSS-syötteen, jotka tarjoavat syötettä, ja lisäisin ne pysyvästi sovellukseen. Tämän jälkeen asiakas voisi asetuksista valita, minkä kunnan tai kuntien RSS-syötteen hän haluaisi. Viimeiseksi lisäisin mahdollisuuden vaihtaa sovelluksen kieltä. Tämä on onneksi tehty Androidille helpoksi ja tallensin kaikki parametrit erilliselle strings.xml-tiedostolle. Luomalla erilliset strings.xml-tiedostot ruotsin ja englannin kielelle koodissa pystyy SharedPreferences-objektilla tallentamaan käyttäjän valinta. Alla on esimerkki, kuinka sovelluksen tekstit tallennetaan strings.xml-tiedostoon. Vaihtamalla <string name=”reloadMenuText”>Päivitä</string> päivitä sanan englanniksi Refresh, niin tämä näkyy sovelluksessa käännettynä ilman, että teimme mitään koodissa.

```
<resources>
<string name="app_name">Kokkola RSS</string>
<string name="settingsMenuText">Asetukset</string>
<string name="validUrlText">Syötä validi Rss feed osoite</string>
<string name="reloadMenuText">Päivitä</string>
<string name="feedChangeText">Vaiha suodatinta:</string>
</resources>
```

LÄHTEET

Androidsuomi.fi. 2016. Mikä on Android? Saatavissa: http://blog.Androidsuomi.fi/mika-on-Android/. Viitattu 14.03.2016.

Statista.com. 2016. Number of available applications in the Google Play Store from December 2009 to November 2015. Saatavissa: http://www.statista.com/statistics/266210/number-of-available-applications-in-the-Google-play-store/. Viitattu 14.03.2016.

Statista.com. 2016. Most popular Google Play app categories in February 2014, by device installs. Kuva. Saatavissa: http://www.statista.com/statistics/279286/distribution-of-worldwide-Google-play-app-downloads-by-category/. Viitattu 14.03.2016.

Source.android.com. 2017. ART and Dalvik. Saatavissa: https://source.android.com/devices/tech/dalvik/. Viitattu 15.05.2017

Developer.Android.com. 2016. Android Studio Overview. Saatavissa: http://developer.Android.com/tools/studio/index.html#build-system. Viitattu 15.03.2016.

Developer.Android.com. 2016. Android Studio Configuration. Saatavissa: http://developer.Android.com/tools/studio/studio-config.html#sdk-mgr. Viitattu 15.03.2016.

Developer.Android.com. 2016. Activities. Saatavissa: http://developer.Android.com/guide/components/activities.html. Viitattu 29.03.2016.

Developer.Android.com.2016. Intents and Intent Filters. Saatavissa: http://developer.Android.com/guide/components/intents-filters.html. Viitattu 29.03.2016.

Developer.xamarin.com. 2016. Understanding Android API Levels. Saatavissa: https://developer.xamarin.com/guides/Android/application_fundamentals/understanding_Android_api_levels/. Viitattu 11.04.2016.

Rssboard.org. 2009. RSS 2.0 Specification. Saatavissa: http://www.rssboard.org/rss-specification. Viitattu 26.04.2016.